

Solution 1: AdaBoost - Updates

- (a) For the first case, only one sample is incorrect. The weighted in-sample misclassification rate can be calculated as:

$$\text{err}^{[m]} = \sum_{i=1}^n w^{[m](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[m]}(\mathbf{x}^{(i)})\}} = 0.01 \quad (1)$$

We can now calculate the weight of the base learner:

$$\hat{\beta}^{[m]} = \frac{1}{2} \log \left(\frac{1 - \text{err}^{[m]}}{\text{err}^{[m]}} \right) = 0.5 \cdot \log \left(\frac{0.99}{0.01} \right) = \log(\sqrt{99}) \quad (2)$$

The updated weights are then calculated as:

$$\begin{aligned} w^{[m+1](i)} &= w^{[m](i)} \cdot \exp \left(-\hat{\beta}^{[m]} \cdot \underbrace{y^{(i)} \cdot \hat{h}(\mathbf{x}^{(i)})}_{\{-1,1\}} \right) \\ &= \begin{cases} 0.01 \cdot \exp(-\log(\sqrt{99}) \cdot 1 \cdot 1) & \text{if } i = 1, \dots, 4, 6, \dots, 10 \\ 0.01 \cdot \exp(-\log(\sqrt{99}) \cdot 1 \cdot -1) & \text{if } i = 5 \\ 0.1 \cdot \exp(-\log(\sqrt{99}) \cdot 1 \cdot 1) & \text{if } i = 11, \dots, 19 \end{cases} \\ &= \begin{cases} 0.001 & \text{if } i = 1, \dots, 4, 6, \dots, 10 \\ 0.0995 & \text{if } i = 5 \\ 0.01 & \text{if } i = 11, \dots, 19 \end{cases} \end{aligned} \quad (3)$$

- (b) In the second case, five out of the 9 samples with weight 0.1 are misclassified. The weighted in-sample misclassification rate can be calculated as:

$$\text{err}^{[m]} = \sum_{i=1}^n w^{[m](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[m]}(\mathbf{x}^{(i)})\}} = 0.5 \quad (4)$$

We can now calculate the weight of the base learner:

$$\hat{\beta}^{[m]} = \frac{1}{2} \log \left(\frac{1 - \text{err}^{[m]}}{\text{err}^{[m]}} \right) = 0.5 \cdot \log \left(\frac{0.5}{0.5} \right) = 0 \quad (5)$$

The updated weights are then calculated as:

$$\begin{aligned} w^{[m+1](i)} &= w^{[m](i)} \cdot \exp \left(-\hat{\beta}^{[m]} \cdot \underbrace{y^{(i)} \cdot \hat{h}(\mathbf{x}^{(i)})}_{\{-1,1\}} \right) \\ &= \begin{cases} 0.01 \cdot \exp(-0 \cdot 1 \cdot 1) & \text{if } i = 1, \dots, 10 \\ 0.1 \cdot \exp(-0 \cdot 1 \cdot -1) & \text{if } i = 15, 16, 17, 18 \\ 0.1 \cdot \exp(-0 \cdot 1 \cdot 1) & \text{if } i = 11, \dots, 14, 19 \end{cases} \\ &= \begin{cases} 0.01 & \text{if } i = 1, \dots, 10 \\ 0.1 & \text{if } i = 15, 16, 17, 18 \\ 0.1 & \text{if } i = 11, \dots, 14, 19 \end{cases} \end{aligned} \quad (6)$$

When the weighted in-sample misclassification rate is 0.5, the weight of the base learner is 0 and we don't do any updates to the weights.

- (c) In the third case, all samples with the exception of one with a weight of 0.01 are misclassified. The weighted in-sample misclassification rate can be calculated as:

$$\text{err}^{[m]} = \sum_{i=1}^n w^{[m](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[m]}(\mathbf{x}^{(i)})\}} = 0.99 \quad (7)$$

We can now calculate the weight of the base learner:

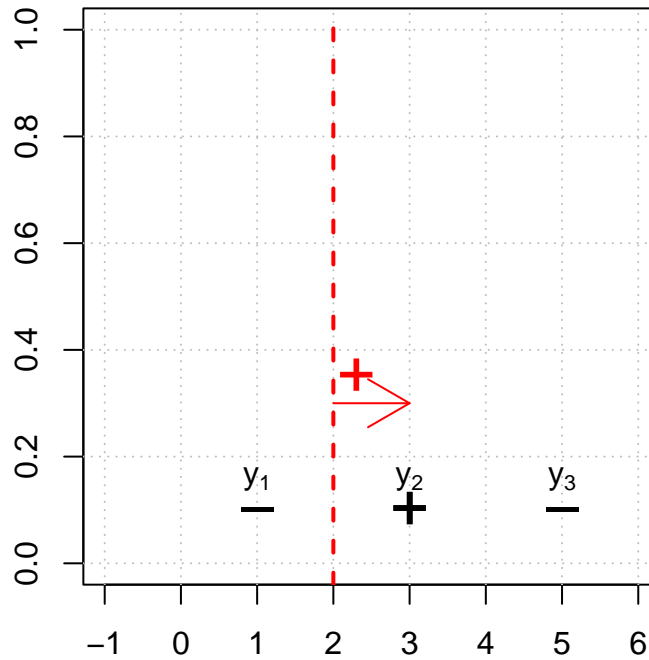
$$\hat{\beta}^{[m]} = \frac{1}{2} \log \left(\frac{1 - \text{err}^{[m]}}{\text{err}^{[m]}} \right) = 0.5 \cdot \log \left(\frac{0.01}{0.99} \right) = \log \left(\sqrt{\frac{1}{99}} \right) \quad (8)$$

The updated weights are then calculated as:

$$\begin{aligned} w^{[m+1](i)} &= w^{[m](i)} \cdot \exp \left(-\hat{\beta}^{[m]} \cdot \underbrace{y^{(i)} \cdot \hat{h}(\mathbf{x}^{(i)})}_{\{-1,1\}} \right) \\ &= \begin{cases} 0.01 \cdot \exp \left(-\log \left(\sqrt{\frac{1}{99}} \right) \cdot 1 \cdot -1 \right) & \text{if } i = 1, \dots, 9 \\ 0.01 \cdot \exp \left(-\log \left(\sqrt{\frac{1}{99}} \right) \cdot 1 \cdot 1 \right) & \text{if } i = 10 \\ 0.1 \cdot \exp \left(-\log \left(\sqrt{\frac{1}{99}} \right) \cdot 1 \cdot -1 \right) & \text{if } i = 11, \dots, 19 \end{cases} \\ &= \begin{cases} 0.001 & \text{if } i = 1, \dots, 9 \\ 0.0995 & \text{if } i = 10 \\ 0.01 & \text{if } i = 11, \dots, 19 \end{cases} \end{aligned} \quad (9)$$

Solution 2: AdaBoost - Decision Stump

- (a) The initial weights all three points in the dataset is $\frac{1}{3}$. A decision boundary for the first decision stump could be the line $x = 2$.



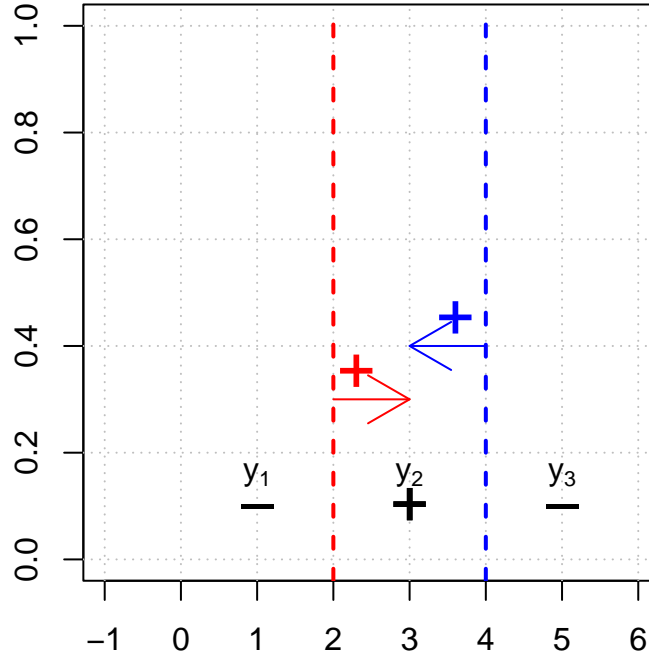
- (b) The first stump makes two correct predictions and one incorrect prediction, we can then calculate the weighted in-sample misclassification rate, the weight for the stump and the new data points weights

$$\begin{aligned}
\text{err}^{[0]} &= \sum_{i=1}^n w^{[0](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[0]}(\mathbf{x}^{(i)})\}} = 0.33 \\
\hat{\beta}^{[0]} &= \frac{1}{2} \log \left(\frac{1 - \text{err}^{[0]}}{\text{err}^{[0]}} \right) = 0.5 \cdot \log \left(\frac{0.67}{0.33} \right) \approx 0.35 \\
w^{1} &= w^{[0](1)} \cdot \exp \left(-\hat{\beta}^{[0]} \cdot y^{(1)} \cdot \hat{b}^{[0]}(\mathbf{x}^{(1)}) \right) = 0.33 \cdot \exp(-0.35 \cdot -1 \cdot -1) \approx 0.23 \\
w^{[1](2)} &= w^{[0](2)} \cdot \exp \left(-\hat{\beta}^{[0]} \cdot y^{(2)} \cdot \hat{b}^{[0]}(\mathbf{x}^{(2)}) \right) = 0.33 \cdot \exp(-0.35 \cdot 1 \cdot 1) \approx 0.23 \\
w^{[1](3)} &= w^{[0](3)} \cdot \exp \left(-\hat{\beta}^{[0]} \cdot y^{(3)} \cdot \hat{b}^{[0]}(\mathbf{x}^{(3)}) \right) = 0.33 \cdot \exp(-0.35 \cdot 1 \cdot -1) \approx 0.47
\end{aligned} \tag{10}$$

We need to normalize the weights so that they sum up to one:

$$\begin{aligned}
w^{1} &= \frac{w^{1}}{\sum_{i=1}^n w^{[1](i)}} = \frac{0.23}{0.23 + 0.23 + 0.47} \approx 0.25 \\
w^{[1](2)} &= \frac{w^{[1](2)}}{\sum_{i=1}^n w^{[1](i)}} = \frac{0.23}{0.23 + 0.23 + 0.47} \approx 0.25 \\
w^{[1](3)} &= \frac{w^{[1](3)}}{\sum_{i=1}^n w^{[1](i)}} = \frac{0.47}{0.23 + 0.23 + 0.47} \approx 0.50
\end{aligned} \tag{11}$$

- (c) As the training error is not yet 0, we do a second stump using the new weights, The decision boundary is $x=4$. The stump makes two correct predictions and one incorrect prediction:



We calculate the weighted in-sample misclassification rate and the weight for the stump.

$$\begin{aligned} \text{err}^{[1]} &= \sum_{i=1}^n w^{[1](i)} \cdot \mathbb{1}_{\{y^{(i)} \neq \hat{b}^{[1]}(\mathbf{x}^{(i)})\}} = 0.25 \\ \hat{\beta}^{[1]} &= \frac{1}{2} \log \left(\frac{1 - \text{err}^{[1]}}{\text{err}^{[1]}} \right) = 0.5 \cdot \log \left(\frac{0.75}{0.25} \right) \approx 0.54 \end{aligned} \quad (12)$$

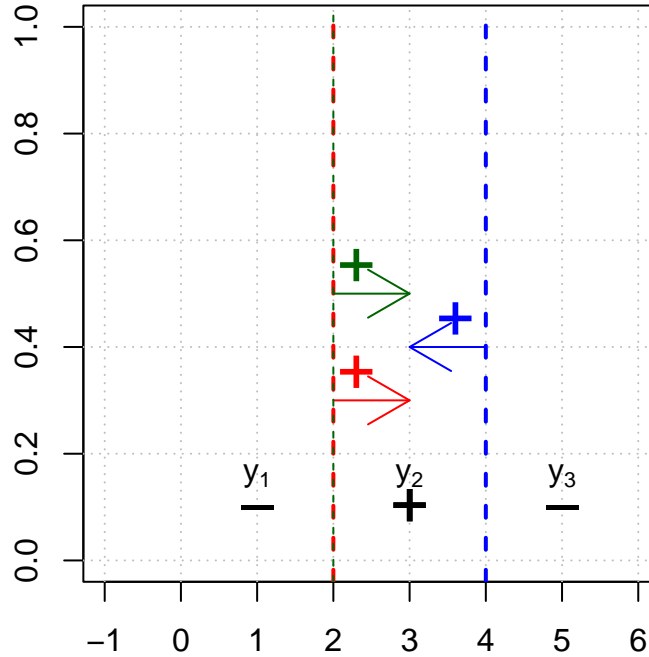
We can see that for the left-most point and the right-most point, there is a disagreement between our two stumps. In the case of the right-most point, the second stump has a bigger weight than the first one, so we will classify it correctly as $y = -1$. Unfortunately, in the case of the left-most point, we will classify it incorrectly as $y = 1$. We will now calculate the new weights:

$$\begin{aligned} w^{[2](1)} &= w^{1} \cdot \exp \left(-\hat{\beta}^{[0]} \cdot y^{(1)} \cdot \hat{b}^{[0]}(\mathbf{x}^{(1)}) \right) = 0.25 \cdot \exp(-0.54 \cdot 1 \cdot -1) \approx 0.42 \\ w^{2} &= w^{[1](2)} \cdot \exp \left(-\hat{\beta}^{[0]} \cdot y^{(2)} \cdot \hat{b}^{[0]}(\mathbf{x}^{(2)}) \right) = 0.25 \cdot \exp(-0.54 \cdot 1 \cdot 1) \approx 0.15 \\ w^{[2](3)} &= w^{[1](3)} \cdot \exp \left(-\hat{\beta}^{[0]} \cdot y^{(3)} \cdot \hat{b}^{[0]}(\mathbf{x}^{(3)}) \right) = 0.5 \cdot \exp(-0.54 \cdot -1 \cdot -1) \approx 0.29 \end{aligned} \quad (13)$$

We need to normalize the weights so that they sum up to one:

$$\begin{aligned} w^{[2](1)} &= \frac{w^{[2](1)}}{\sum_{i=1}^n w^{[2](i)}} = \frac{0.42}{0.42 + 0.15 + 0.29} \approx 0.49 \\ w^{2} &= \frac{w^{2}}{\sum_{i=1}^n w^{[2](i)}} = \frac{0.23}{0.42 + 0.15 + 0.29} \approx 0.17 \\ w^{[2](3)} &= \frac{w^{[2](3)}}{\sum_{i=1}^n w^{[2](i)}} = \frac{0.47}{0.42 + 0.15 + 0.29} \approx 0.34 \end{aligned} \quad (14)$$

We will start with a third iteration that will correct this mistake. Unfortunately, we can't have open end notes in a stump during training. The best we can do add a new stump in the same place as the first one:



We calculate the weighted in-sample misclassification error for this stump:

$$\begin{aligned}\text{err}^{[1]} &= \sum_{i=1}^n w^{[1](i)} \cdot \mathbb{1}_{\{y^{(1)} \neq \hat{b}^{[1]}(\mathbf{x}^{(i)})\}} = 0.34 \\ \hat{\beta}^{[1]} &= \frac{1}{2} \log \left(\frac{1 - \text{err}^{[1]}}{\text{err}^{[1]}} \right) = 0.5 \cdot \log \left(\frac{0.66}{0.34} \right) \approx 0.54\end{aligned}\tag{15}$$

We can see that we are going back to a similar case as the first iteration and now the rounding errors could play a role in our calculations. Let's build a model to see what happens:

```
df = data.frame(x1,x2,y)
for (i in 1:6){
  # Train the model with the amount of iterations
  model= sboost::sboost(df[1:2],df$y,iterations = i)
  # Assess the performance
  model_assessment=sboost::assess(model,features = df[1:2],outcomes = df$y)
  cat("Iteration number : " , i,
      ". model accuracy: ", model_assessment$performance[[6]],"\n" )
}

## Iteration number : 1 . model accuracy:  0.6666667
## Iteration number : 2 . model accuracy:  0.6666667
## Iteration number : 3 . model accuracy:  0.6666667
## Iteration number : 4 . model accuracy:  0.6666667
## Iteration number : 5 . model accuracy:  0.6666667
## Iteration number : 6 . model accuracy:  0.6666667

cat("Voting power of each stump: \n", model$classifier$vote)

## Voting power of each stump:
##  0.3465736 0.5493061 0.3465736 0.2554128 0.2027326 0.1682361

cat("Where each stump is located: ", model$classifier$split )

## Where each stump is located:  2 4 2 4 2 4
```